Latent space exploration and functionalization of a gated working memory model using conceptors

Anthony Strock, Nicolas P. Rougier, Xavier Hinaut

Latent space exploration and functionalization of a gated working memory model using conceptors

Anthony Strock[2,1,3], Nicolas P. Rougier[1,2,3,†] and Xavier Hinaut[1,2,3,†,*]

[1]Inria Bordeaux Sud-Ouest, Talence, France
[2]LaBRI, Université de Bordeaux, CNRS UMR 5800, Talence, France
[3]IMN, Université de Bordeaux, CNRS UMR 5293, Bordeaux, France
† Equal contribution, *Corresponding author: xavier.hinaut()inria.fr

Abstract

Introduction. Working memory is the ability to maintain and manipulate information. We introduce a method based on conceptors that allows us to manipulate information stored in the dynamics (latent space) of a gated working memory model. **Methods.** This latter model is based on a reservoir: a random recurrent network with trainable readouts. It is trained to hold a value in memory given an input stream when a gate signal is on and to maintain this information when the gate is off. The memorized information results in complex dynamics inside the reservoir that can be faithfully captured by a conceptor. **Results.** Such conceptors allow us to explicitly manipulate this information in order to perform various, but not arbitrary, operations. In this work, we show (1) how working memory can be stabilized or discretized using such conceptors, (2) how such conceptors can be linearly combined to form new memories, and (3) how these conceptors can be extended to a functional role. **Conclusion.** These preliminary results suggest that conceptors can be used to manipulate the latent space of the working memory even though several results we introduce are not as intuitive as one would expect.

1 Introduction

A recent and major enhancement of the Reservoir Computing (RC) paradigm has been proposed by Jaeger (2014) and Jaeger (2017) under the form of *conceptors* which are able to capture the subspace of internal states of a Recurrent Neural Network (RNN). In the case of Echo State Networks (ESN), conceptors can be used to capture the trajectory of reservoir states when the reservoir is fed with a particular input pattern (see figure 1). These conceptors allow to extend the capacity of the original ESNs by taking advantage of these new representations. For example, (Jaeger, 2014; Bao et al., 2016; Bartlett et al., 2019; Gast et al., 2017) showed how to use them for the recognition of temporal sequences while using them for the storage and retrieval of multiple temporal sequences. More recently, Mossakowski, Diaconescu, and Glauer (2019) proposed an implementation of fuzzy logic based on conceptors, while Liu, Ungar, and Sedoc (2019) used conceptors for online learning of sentence representations, and He and Jaeger (2018) proposed a general way to use conceptors during the learning of multiple tasks.

Conceptors yield several advantages when compared to classical reservoirs since Jaeger (2014) demonstrated that conceptors can be used for performing symbolic operations in the latent space of the different input patterns. Such symbolic operations have been already exploited in the framework of deep learning community and they provide impressive results. For instance, in natural language processing (NLP), Mikolov et al. (2013) showed that arithmetic operations such as "$king - men + woman$" give a vector similar to "$queen$". More recently, Brock et al. (2016) proposed a method to edit global image features based on operations performed on the latent space of generative adversarial networks (GANs). Conceptors provide similar logical operations in the framework of the reservoir computing paradigm. For instance, Jaeger (2014) proposes an operator that quantifies if a stimulus is similar to an already known conceptor. By associating one conceptor per class, it is possible to measure if a stimulus belongs to a class (positive evidence) or none (negative evidence). Beyond logical operations, linear combinations of conceptors allow to implement continuous morphing between set of states: they were used to create morphing between two time series corresponding to the extended interpolation of the time series (e.g. a morphing between two sine-waves with different frequencies is a sine-wave with an intermediate frequency).

This capacity of performing operations in the latent space resonates strongly with the notion of working memory (WM) as found in neuroscience. It is generally defined as the capacity to hold information for a short period of time as well as the capacity to manipulate this information in order to achieve some task or to reach a specific goal. In this context, we have introduced in (Strock, Hinaut, and Rougier, 2020) a reservoir with feedback connections that implements a gated working memory, i.e. a generic mechanism to maintain information at a given time (corresponding to when the gate is on, see figure 3). In this model, the memory is encoded in the dynamics of the reservoir and information can be maintained without any sustained activity. This absence of sustained activity is precisely what makes it difficult to manipulate the underlying information and this is also the reason why some authors (Mongillo, Barak, and Tsodyks, 2008; Stokes, 2015; Masse et al., 2019; Manohar et al., 2019) have suggested the existence of a mechanism to temporarily store information in synaptic weights. In this context, conceptors provide a plausible explanation for such a transfer as well as an explicit method for manipulating information; even if the conceptor mechanisms are for the moment not as biologically plausible as the reservoirs themselves.

In this article, we explore the nature of operations carried on by such conceptors and explore the different ways to combine them such as to explicitly manipulate memories. Even though the results we

introduce in this article are preliminary and to some extents, counter-intuitive, this leads us to consider the notion of functional conceptors that would allow to arbitrarily manipulate working memory in the latent space.

2 Methods

2.1 Conceptors overview

Considering an ESN R that has been trained[1] to produce the sequence O_1 when presented with input sequence I_1, Jaeger (2014) demonstrated that it is possible to build an ESN $R*$ that will spontaneously produce the sequence O_1, in the absence of any input (see Figure 1). The activity of this new $R*$ can be decoded using the read-out weights of R. This is actually similar to the principle of the full-FORCE method introduced in (DePasquale et al., 2018) where internal weights are trained to match the internal activity of a teacher network receiving the desired output as input. However, Jaeger (2014) principle is applicable to multiple input patterns with the assumption that each input pattern makes the reservoir evolve in a separable region of the internal high dimensional space. In order to build $R*$, he proposed to approximate the activity of R when it receives any of these input patterns. Then he equips $R*$ with a set of recurrent connections (i.e. the conceptors) that are specific to each couple of input/output and that will project the internal state into the relevant sub-space. Jaeger (2017) shows in particular how these conceptors can be considered as long-term memories for temporal patterns. Conceptors can store temporal patterns and reactivate them later with negligible loss of recall/precision. More generally, conceptors can be considered as long-term memories of internal states subspaces.

2.2 Models

2.2.1 Echo State Networks (ESN)

In this work we consider Echo State Networks (ESN) with feedback from readout units to the reservoir (Jaeger, 2001). The system is described by the following update equations:

$$\begin{aligned} x[n] &= \tanh\left(W_{in}u[n] + Wx[n-1] + W_{fb}y[n-1]\right) + \xi \\ y[n] &= W_{out}x[n] \end{aligned}$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb} and W_{out} are respectively the recurrent, the input, the feedback, the output weight matrices and ξ is a uniform white noise term added to reservoir units.

2.2.2 Controlling ESN dynamics using a conceptor

Following (Jaeger, 2014) notations, the equation for a conceptor C enforcing some particular dynamics can be written as:

$$x[n] = C\tanh\left(Wx[n-1] + b\right)$$

where C is the conceptor (possibly changing over time), $x[n]$ is the state of the model at time n, W is the recurrent matrix and b is a constant bias. This can be extended to the general case where we also have an input $u[n]$ (with input matrix W_{in}) (or similarly a feedback), and writes:

$$\begin{aligned} x[n] &= C\tanh\left(Wx[n-1] + W_{in}u[n] + W_{fb}y[n-1]\right) \\ y[n] &= W_{out}x[n] \end{aligned}$$

[1] offline with ridge regression

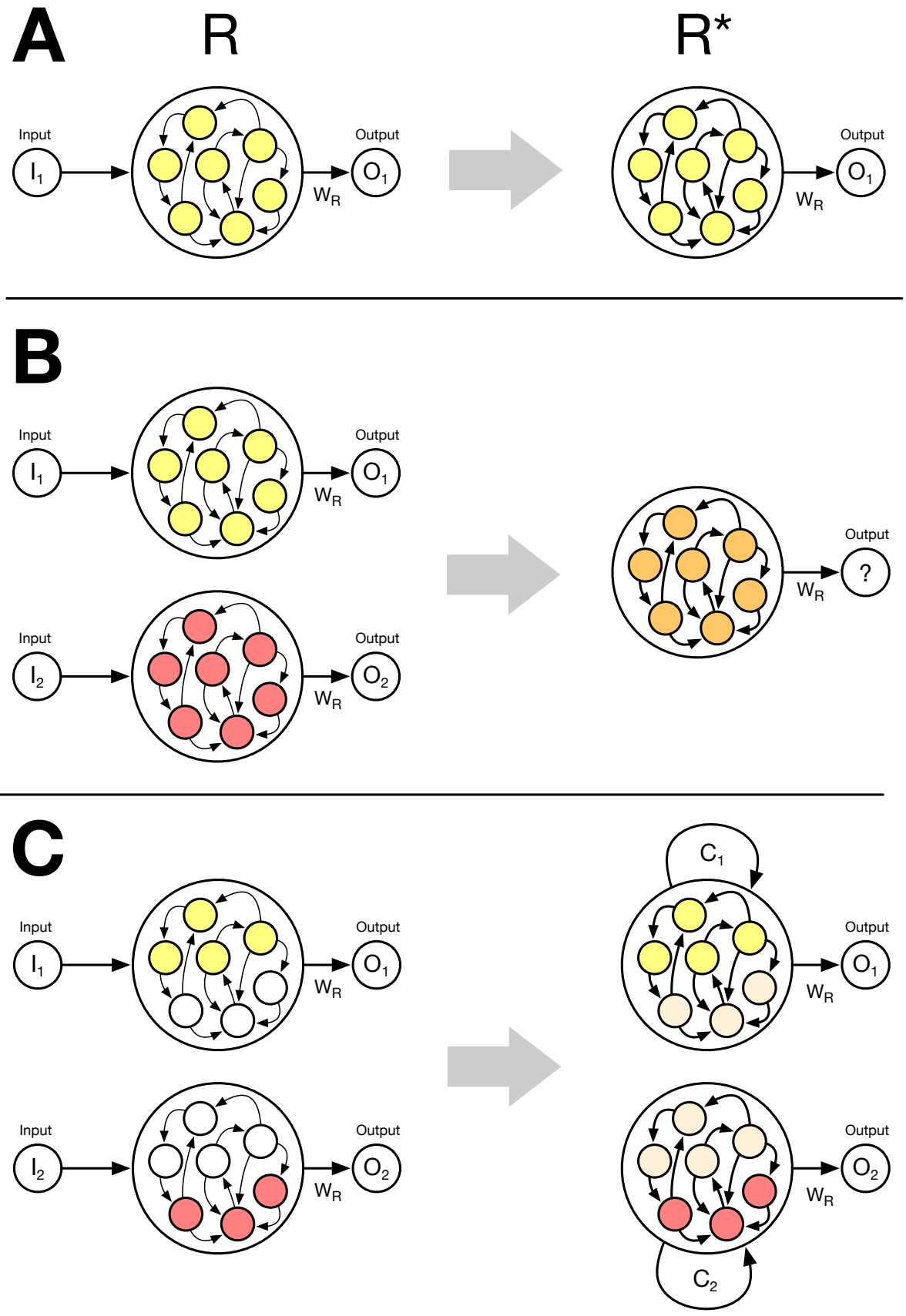

Figure 1. Conceptors general idea A Considering an ESN R that outputs the sequence O_1 when an input I_1 is presented, it is possible to build an ESN $R*$ that spontaneously outputs the sequence O_1 in the absence of any input. **B** Considering an ESN R that respectively outputs the sequences O_1 and O_2 when input I_1 and I_2 are presented, it is not possible to define $R*$ as in **A** since we cannot define the expected output in the absence of input. **C** Considering an ESN R that respectively outputs the sequences O_1 and O_2 when input I_1 and I_2 are presented – with the supplementary conditions that the inner representation corresponding to inputs I_1 and I_2 are separable – it is possible to build an ESN $R*$ equipped with a set of feedback weights C_1 or C_2 such that

Using a conceptor C is similar to a change of W in $\tilde{W} = WC$ (and W_{out} in $W_{\text{out}}C$ if there is feedback). In our implementation, we thus consider:

$$x[n] = \tanh\left((W + W_{fb}W_{out})\ C\ x[n-1] + W_{in}u[n]\right)$$
$$y[n] = W_{out}x[n]$$

2.2.3 Computing conceptors

In order to compute a conceptor for some given dynamics, it is necessary to collect all the states of the reservoir and to concatenate them in a matrix X. The conceptor C is then defined as:

$$C = XX^T\left(XX^T + \alpha^{-2}I\right)^{-1} = R\left(R + \alpha^{-2}I\right)^{-1}$$

where $R = XX^T$ is similar to a covariance matrix, and α (a.k.a the aperture) controls how close from the identity matrix C is.

2.2.4 Aperture adaptation

Intuitively, the aperture of a conceptor controls the precision of the internal states representation. However, no information on internal states is lost, because it is possible to change the aperture of a conceptor C without the need to recompute the conceptor from scratch. To change the aperture, one only need to adapt the conceptor C as follows:

$$\phi(C, \gamma) = C\left(C + \gamma^{-2}(I - C)\right)^{-1}$$

where $\phi(C, \gamma)$ represents the same states than C with a different aperture, and γ is controlling how the aperture is modified. Intuitively, $\phi(C, \gamma)$ modifies the aperture of C by a factor of γ.

2.2.5 Linear combination

Given two conceptors C and B and $\lambda \in \mathbb{R}$, the linear combination of conceptor C and B is defined as:

$$C = \lambda C + (1 - \lambda)B$$

In the following when $\lambda \in [0, 1]$ we will talk about interpolation, when $\lambda > 1$ about right-extrapolation, and when $\lambda < 0$ about left-extrapolation.

2.2.6 Boolean operations

Boolean operations can be written as:

$$C \vee B = \left(I + \left(C(I - C)^{-1} + B(I - B)^{-1}\right)^{-1}\right)^{-1}$$
$$C \wedge B = \left(C^{-1} + B^{-1} - I\right)^{-1}$$
$$\neg C = I - C$$

However, as highlighted in (Mossakowski, Diaconescu, and Glauer, 2019), $\vee$ and $\wedge$ are not idempotent (i.e. $C \vee C \neq C$ and $C \wedge C \neq C$). More precisely if C (resp. B) is a conceptor built with the covariance matrix R (resp. Q), Jaeger proposes to build $C \vee B$ using the covariance matrix $R + Q$ that is by design not idempotent. What we propose here is to consider instead the matrix $\beta R + (1 - \beta)Q$ with $\beta \in [0, 1]$

instead of $R + Q$, or if we want it to be symmetric $(R + Q)/2$. Similar calculation gives the following new $\vee_\beta$ and $\wedge_\beta$.

$$C \vee_\beta B = \left(I + \left(\beta C(I - C)^{-1} + (1 - \beta)B(I - B)^{-1}\right)^{-1}\right)^{-1}$$
$$C \wedge_\beta B = \left(\beta C^{-1} + (1 - \beta)B^{-1}\right)^{-1}$$

This way of building the OR operation also has a data driven intuition. If we note $\beta = \frac{n}{n+p}$ where n (resp. p) is the number of data points used to build R (resp. Q) then $bR + (1 - b)Q$ is the "correlation matrix" obtained by taking the union of all the data points. Moreover, if we choose $\beta = 0.5$ then there is a direct link between the two ways of defining the OR operator: $C \vee B = \phi(C \vee_{0.5} B, 2)$. In this study, the aperture was mostly not influencing the results, thus we show only the results for $\vee$.

2.3 Gating task

We consider the gating task described in (Strock, Hinaut, and Rougier, 2020). In this task the model receives an input V that is continuously varying over time and another input being either 0 or 1 (trigger or gate T). To complete the task, the output has to be updated to the value of the input when the trigger is active and to remain constant otherwise (similarly to a line attractor). In other words, the trigger acts as a gate that controls the entry of the value in the memory (the output). Figure 2 describes this task.

Figure 2. Gating task. Each column represents a time step (time increases from left to right), colored discs represent inputs (V and T) and the output (M).

2.3.1 Gated Working Memory Reservoir

We consider a reservoir with feedback from readout units to the reservoir (see Figure 3). In (Strock, Hinaut, and Rougier, 2020) we showed that this gated working memory reservoir is able to learn to robustly gate information in presence of noise and of a distracting input. The model behaves as a *line attractor* even if few values are used for training (about 10 values is enough). We also provided a minimal model version and showed an equivalence with GRU (Gated Recurrent Unit) cells (Cho et al., 2014), which are a simplified version of Long-Short Term Memory (LSTM) cells.

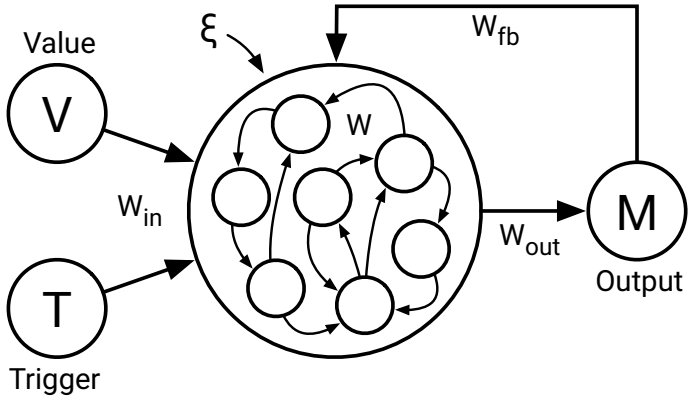

Figure 3. The Gated Working Memory Reservoir model The reservoir receives a random signal V in $[-1, +1]$ and a trigger signal T in $\{0, 1\}$. The output M is fed back to the reservoir.

2.4 Implementation details

We consider a reservoir of 1000 neurons that has been trained to solve a gating task described in Figure 2 **A**. The overall dynamics of the network we consider are described by the following equations:

$$x[n] = \tanh\left(\left(W + W_{fb}W_{out}\right) C\, x[n-1] + W_{in}u[n]\right) + \xi$$
$$y[n] = W_{out}x[n]$$

where $u[n]$, $x[n]$ and $y[n]$ are respectively the input, the reservoir and the output at time n. W, W_{in}, W_{fb}, W_{out} and C are respectively the recurrent, the input, the feedback, the output and the conceptor weight matrices and ξ is a uniform white noise term added to reservoir units. W, W_{in}, W_{fb} are uniformly sampled between -1 and 1, and left untrained. Only W is scaled to have sparsity level equal to 0.5 and a spectral radius of 0.1. If not stated otherwise, the noise is selected uniformly between -10^{-4} and 10^{-4}.

The major difference with Jaeger's proposal in the way the patterns are stored is that in our case patterns are stored implicitly when solving the gating task. In other words, the patterns are stored by training W_{out} and not by explicitly recomputing an internal weight matrix. However as mentioned in (DePasquale et al., 2018), training W_{out} when there is a feedback is equivalent to recomputing the internal weight matrix W as $W^* = W + W_{fb}W_{out}$.

When W_{out} is computed to solve the gating task, the conceptor C is considered to be fixed and equal to the identity matrix ($C = I$). W_{out} is trained for 25,000 time steps. At each time step there is a 0.01 probability of having a trigger and the input value (V) is uniformly randomly sampled between -1 and 1. During training, the average holding time of the value in memory is therefore about 100 time steps.

After training, in normal mode, the conceptor C is equal to a conceptor C_m that is generated and associated to a constant value m. In order to compute this conceptor C_m, we impose a trigger ($T = 1$) as well as an input value ($V = m$) at the first time step, such that the reservoir has to maintain this value for 100 time steps. During these 100 time steps, we use the identity matrix in place of the conceptor. The conceptor C_m is then computed according to $C_m = XX^T\left(XX^T + \frac{I}{a}\right)^{-1}$, where X corresponds to the concatenation of all the 100 reservoir states following the trigger, each row corresponding to a time step, I the identity matrix and a the aperture. In all the experiments the aperture has been fixed to $a = 10$. For the conceptors pre-computed in Figure 4 and 6, the reservoir have been initialised with its last training state.

3 Results

3.1 Constant-memory conceptors

The idea behind what we named *constant-memory conceptors* is to capture explicitly the dynamics of a reservoir maintaining a value and to use this conceptor to later constrain the dynamics of any reservoir, inducing an alternative memory in the output as represented on figure 4. In order to build a constant-memory conceptor C_m, we consider the gated reservoir working memory model that receives a trigger and an input value m at time $t = 0$. We collect the states of the reservoir for 100 iterations from which we build the conceptor C_m and apply it immediately to the model. Results of this procedure is shown on figure 4B where five conceptors are built (gray bands) and applied immediately (white bands) without noticeable modification on the output of the reservoir since the actual dynamics and the dynamics stored in the conceptor are congruent. On figure 4A, we used the same procedure to build a set of 11 conceptors whose captured dynamics correspond to 11 values uniformly spread between -1 and +1. On figure 4C, after 100 iterations following the presentation of a new value, we apply the closest conceptor (Frobenius norm, inducing a distance between conceptors) from the previous set of

11 constant-memory conceptors. One can see that on figure 4C that the input value is first maintained (grey band) and jumps rapidly to the closest discretized value when the conceptor is actually applied and constrained the dynamics.

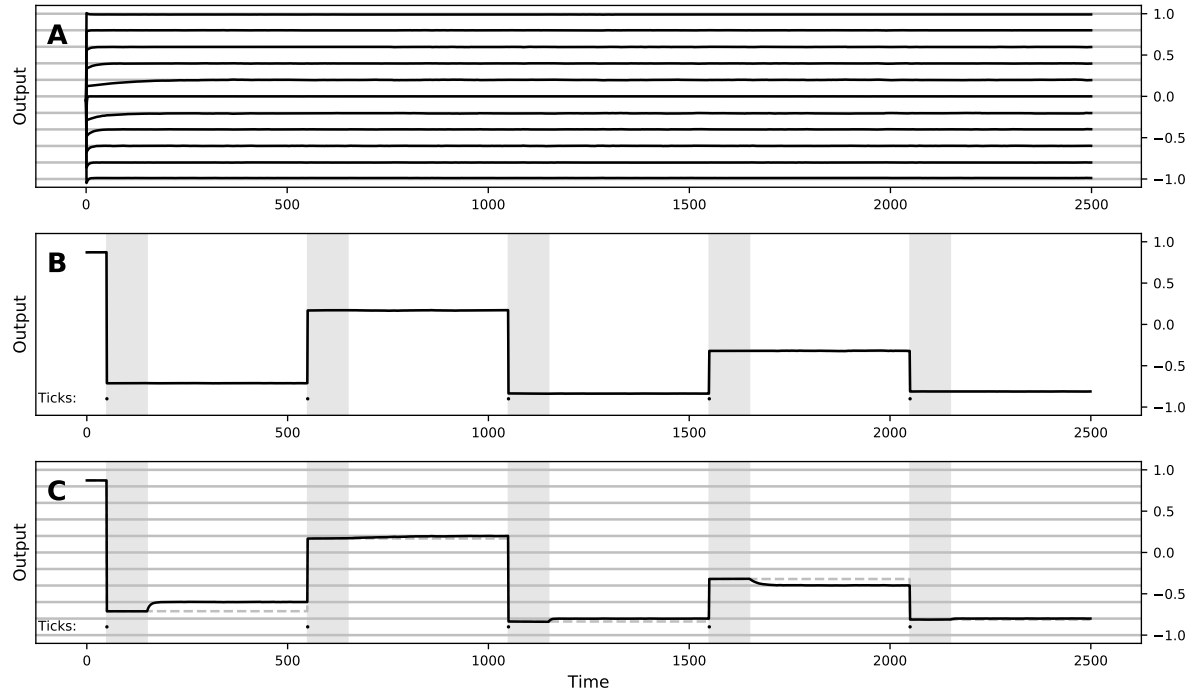

Figure 4. Approximation with conceptors and discrete conceptors. In **A**, **B** and **C** the model receives the same inputs across time. Black lines: Evolution of the reservoir readout y. Each black line in **A.** represents a different trial where a different conceptor is applied. Gray lines: the discrete value considered for each conceptor. Light gray areas: time period when conceptors are computed for the current value. **A.** Different trials showing various discrete conceptors applied. **B**-**C.** Conceptors C_m are computed using the 100 time steps following a trigger while $C = I$ (light gray areas). **B.** The conceptor C_m is directly applied during the 400 following time steps. **C.** The closest conceptor among the discretized conceptor is applied during the 400 following time steps. Dashed lines represents the memory that should have been kept if not discretized.

These constant-memory conceptors also exhibit a nice property regarding the long term maintenance of a memory. The initial gated working memory model is already quite robust regarding the long term maintenance of memory in the absence of internal noise (i.e. inside the reservoir). When it is trained for a few hundreds of time steps, it is able to maintain a memory for several thousands of time steps (see figure 5A) before the memory starts to slowly degrade. When conceptors are applied, this slow degradation vanishes (figure 5B): the RMSE without conceptor is of 4.21e-02 ± 1.84e-02 (mean ± std) whereas with conceptors it is 1.02e-03 ± 6.95e-04. In the presence of internal noise (inside the reservoir), the initial gated working memory model is much less robust and memory starts to degrade after only a few thousand time steps (see figure 5C). More precisely, a 10^{-4} noise ($std(\xi) = 10^{-4}$) prevents the model to maintain a value for a much longer time than the time that has been used to train it (figure 5C). However, when conceptors are applied, their benefit is even more obvious: after one hundred thousand time steps, without conceptor the memory converge towards a few values, whereas with conceptors the memory remains(figure 5D). The RMSE without conceptor becomes 1.39e-01 ± 7.66e-02 (3.3 times greater than without noise), whereas the RMSE with conceptors becomes 2.85e-03 ± 1.98e-03 (2.8 times greater than without noise).

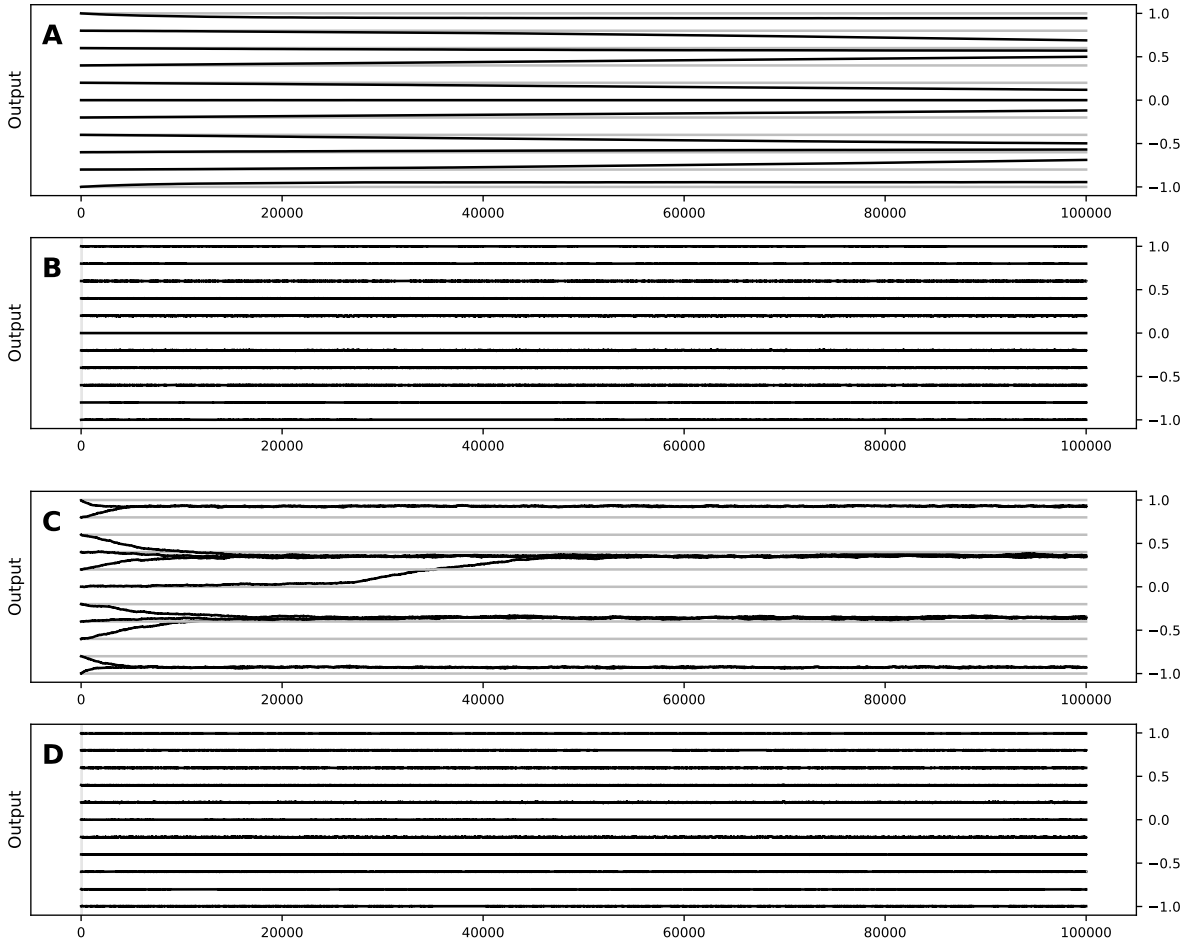

Figure 5. Stability comparison with or without conceptor, with or without noise. Black: Evolution of the readout y. lines: the discrete value considered. Light areas: time when conceptors are computed for the current value. **A**-**B** No noise. **C**-**D** 0.0001 noise. **A** and **C** No conceptor used. **B** and **D** Discrete conceptor are applied.

3.2 Linear interpolation of constant-memory conceptors

In Figure 6, we show two main ideas: **(1)** how a linear interpolation between two conceptors can allow to generalize the gating of other values, and **(2)** a representation of the space in which lies the conceptors and their link to the memory they encode. **(1)** Interpolation and extrapolation C of conceptor $C_{0.1}$ and conceptor $C_{1.0}$ has been computed as $C = \lambda C_{1.0} + (1 - \lambda)C_{0.1}$ with 31 λ values uniformly spread between -1 and 2. Even though the interpolated ($\lambda \in [0, 1]$) conceptors obtained are not exactly equivalent to C_m conceptors obtained in Figure 4, they seem to also correspond to a retrieved memory value that is maintained. The mapping between λ and the value is non-linearly encoded. For right-extrapolation ($\lambda \in [1, 2]$) the conceptor seems to be linked to a noisy version of a C_m conceptor: the output activity is not constant, but its moving average is constant. For left-extrapolation ($\lambda \in [-1, 0]$), the conceptor obtained does not seem to encode any information anymore: all the output activities collapse to zero. **(2)** Principal Component Analysis (PCA) have been performed using 201 pre-computed conceptors associated to values uniformly spread between -1 and 1. The first three components already explain approximately 85% of the variance. The first component seems to non-linearly encode the absolute value of the memory (Figure 6B) whereas the second component seems to non-linearly encode the memory itself (Figure 6C). The curved line composed of conceptors C_m (Figure 6E-G) shows visually why the extrapolation does not work as we expected.

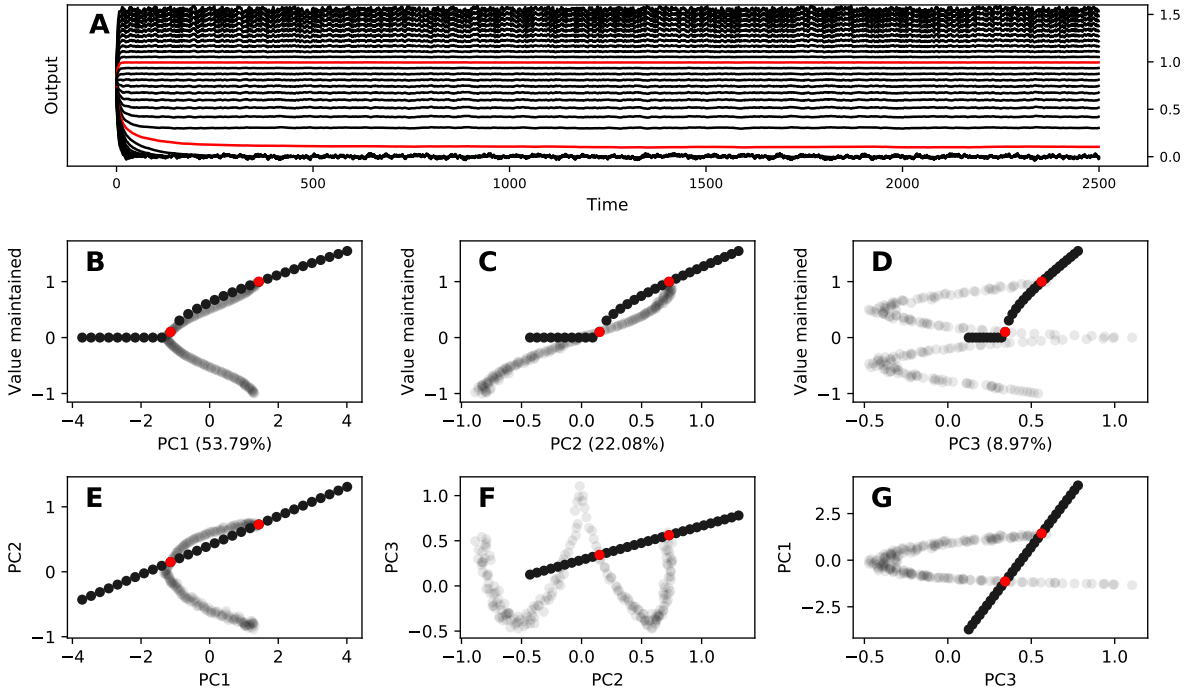

Figure 6. Generalisation of constant-memory conceptors C_m. **Red**: two constant-memory conceptors: $C_{0.1}$ and $C_{1.0}$. **Black**: Inferred conceptors, i.e. linear interpolation and extrapolation between $C_{0.1}$ and $C_{1.0}$. **A** Temporal evolution of the readout for different conceptors. **B**-**G** : constant-memory conceptors C_m for 201 values of m uniformly spread between -1 and 1. **B**-**D** Link between principal components of the conceptors and the memory they are encoding. For the interpolated conceptors, the memory is considered as the mean in the last 1000 time steps. **E**-**G** Representation of the conceptors in the three principal components of the C_m conceptors.

3.3 Functional conceptors

In this section, we show three examples where conceptors have a functional role: (1) constant-memory conceptors when triggers are received, (2) a conceptor enforcing triggers, and (3) the con-

junction/disjunction of constant-memory conceptors. By *functional* we mean that conceptors do not only store and reactivate a dynamical pattern. Conceptors can do much more than constrain the dynamics in an attractor. Conceptors can also constrain the dynamics in a space where the behavior is input-dependent, i.e. the outputs are some function of the inputs.

Constant-memory conceptors when triggers are received. When a constant-memory conceptor C_m is applied to the gated working memory model, it seems to quickly force the output y to match the value m, making it insensitive to the input value. However, the actual behavior is a bit different as illustrated on figure 7. On this figure, we can observe that the readout value of the resevoir under the influence of a conceptor $C_{0.5}$ is destabilized when a trigger occurs. More precisely, the trigger induce an instanteneous readout equal to the input until quickly relaxing to the constant value of the conceptor (0.5). This confirms that the combination of a reservoir and a constant-memory conceptor remains sensitive to the input. Said differently, a constant-memory conceptor C_m is not the mere storage of the value m but rather a function that constrains the dynamics of the reservoir such that when applied, the readout will eventually read m after some time.

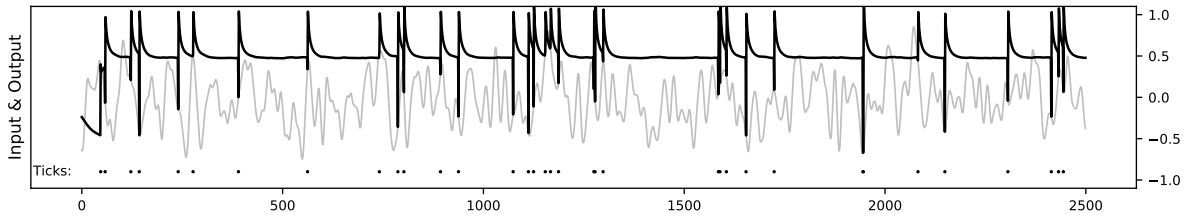

Figure 7. Influence of trigger with constant-memory conceptors. A constant-memory conceptor $C_{0.5}$ is applied while receiving several triggers. Gray: the input value (V). Black: Evolution of the readout y. When a trigger occurs (indicated by dots on the *Ticks* line) the readout transitorily goes to the current input value before going back to the value memorized by the conceptor.

Conceptor enforcing triggers. This functional aspect of conceptor can be further illustrated by building the following conceptor C_T: during 100 time steps, the reservoir receives constant triggers (T = 1) and has therefore to follow the value (V) it receives as input. Conceptor C_T is constructed from the states taken by the reservoir during these 100 time steps. Figure 8 shows what happens when this conceptor is subsequently applied: independently of the trigger signal, the output of the reservoir follows the input. Everything happens as if the reservoir was receiving a continuous trigger signal and the conceptor acts as a pass-through filter that modifies the behavior of the gated working memory as a whole (instead of modifying each individual value). This result suggests that conceptors may probably be extended to store arbitrary functions that act in the latent space of the reservoir. We do not know yet how to specify such arbitrary functions inside a conceptor, but the preliminary results we introduced are encouraging even though more theoretical work is needed.

Conjunction/disjunction of constant-memory conceptors. The last example where we show this functional aspect of conceptors is the disjunction of conceptors. If the conceptors C_m and $C_{m'}$ represent the subspaces when the value stored are respectively m and m', the disjunction of C_m and $C_{m'}$, i.e. $C_m \vee C_{m'}$ would represent the union (or the sum) of such subspaces. Applying $C_m \vee C_{m'}$ might therefore constrain the dynamics in one of these two subspaces, enforcing the memory (i.e. the output y) to become either m or m'. $C_m \vee C_{m'}$ could thus store a choice function between m and m', or in other words, a sort of conditional assignment that would store m in some cases and m' in some others. It is not exactly what $C_m \vee C_{m'}$ does but it still implements some conditional assignment. When

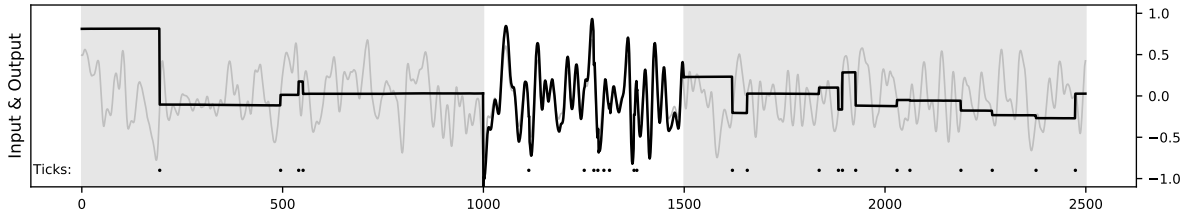

Figure 8. Pass-through conceptors. This conceptor is able to modify the behavior of the gated working model by letting all the values to pass through the reservoir up to the output, independently of the gating signal (time steps 1000 to 1500). Black: Evolution of the readout y. Gray: the input value (V). Light gray areas: no conceptor is applied (i.e. $C = I$). White areas: the conceptor simulating a constant trigger is applied (i.e. $C = C_T$), thus the input is redirected to the output.

a trigger occurs the output jumps towards the value v to be maintained and then relaxes to another value that depends on v.

In Figure 9, we show the values towards which the output relaxes (i.e. relaxation values) when the disjunction of two constant-memory conceptors is applied. First, as the disjunction of twice the same conceptor $C_m \vee C_m$ is either equivalent to the same conceptor or to an aperture adaptation of it (i.e. $C_m \vee C_m = \phi(C_m, 2)$ and $C_m \vee_\beta C_m = C_m$), the value towards which the disjunction $C_m \vee C_m$ relaxes is the value of the conceptor itself (i.e. m). Then, we realized that we could predict what would be the relaxation values in different cases: in general the relaxation value was mostly either almost zero or the maximum of the absolute values of the two conceptors multiplied by the sign of the new value to be maintained. We propose the following formula to predict the value towards which $C_{m_1} \vee C_{m_2}$ relaxes:

$$v_{\text{relax}}(m_1, m_2, v) = \begin{cases} m_1 & \text{if } m_1 = m_2 \\ sign(v) \times max(|m_1|, |m_2|) & \text{if } min(|m_1|, |m_2|) < |v| \\ & \text{or } m_1 = -m_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where v is the initial value (V) proposed along with the trigger, m_1 (resp. m_2) is the constant associated to conceptor m_1 (resp. m_2), $v_{\text{relax}}(m_1, m_2, v)$ is the ultimate value reached while applying conceptor $C_{m_1} \vee C_{m_2}$. Said differently, the conceptor $C_{m_1} \vee C_{m_2}$ implements a conditional assignment (modulo the sign): if the input value at time of trigger is bigger than the minimum between m_1 and m_2 then it will converge towards the maximum of m_1 and m_2, otherwise it will converge towards 0. The predictions made by the formula are less accurate for extreme values such as for $v = 1.00$ (see Figure 9).

We hypothesize a similar formula for relaxation values of n constant-memory conceptors:

$$v_{\text{relax}}(m_1, ..., m_n, v) = \begin{cases} m_1 & \text{if } m_1 = ... = m_n \\ sign(v) \times max(|m_1|, ..., |m_n|) & \text{if } min(|m_1|, ..., |m_n|) < |v| \\ & \text{or } (\forall i, j \ |m_i| = |m_j| \\ & \text{and } \exists i, j \text{ such that } i > j \text{ and } m_i = -m_j) \\ 0 & \text{otherwise} \end{cases}$$

where v is the initial value (V) given along with the trigger, m_i is the constant associated to conceptor C_{m_i}, $v_{\text{relax}}(m_1, m_2, ..., m_n, v)$ is the ultimate value reached while applying conceptor $\bigvee_{i=1}^{n} C_{m_i}$.

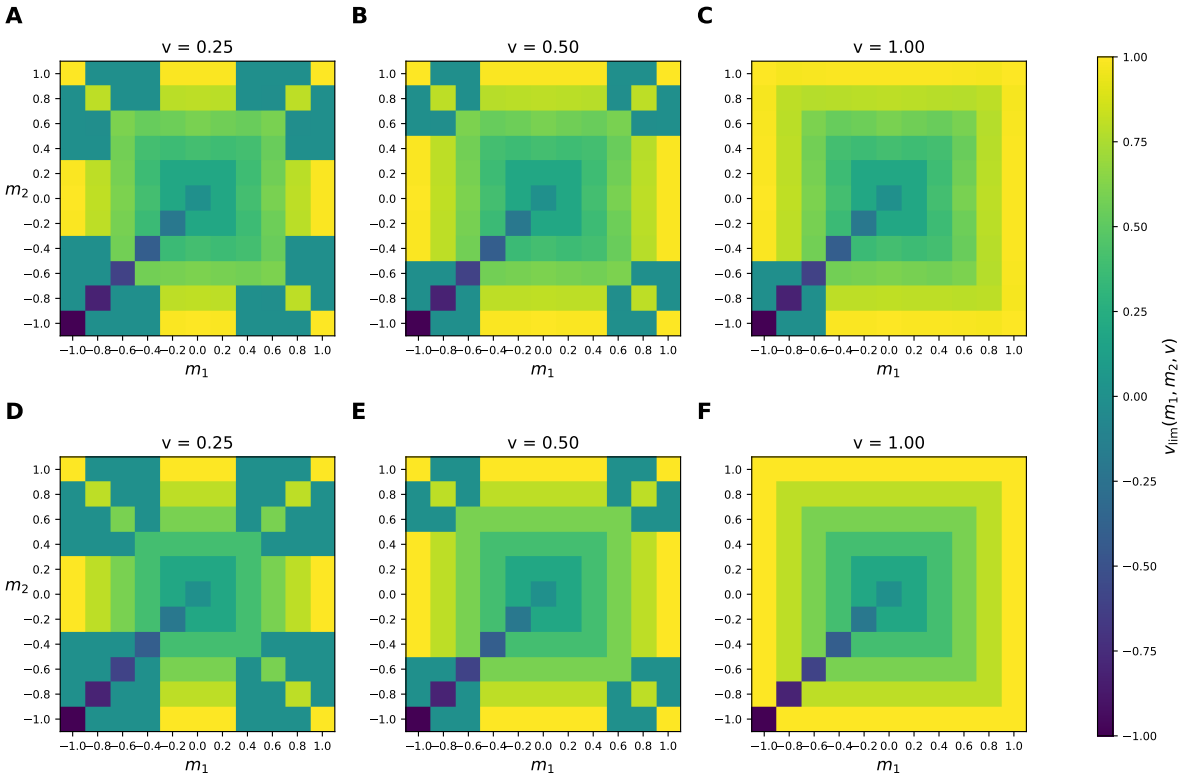

Figure 9. Relaxation values (i.e. values towards which the output converges) when applying the disjunction of two constant-memory conceptors. In other words, it corresponds to the final values reached when applying the conceptor $C_{m_1} \vee C_{m_2}$. **A**-**C** Empirical results from experiments. **D**-**F** Predictions based on equation 1.

We also studied how the conjunction constant-memory conceptors were influencing the dynamics. If the disjunction of conceptors is similar to a union (or sum), then the conjunction of conceptors is similar to an intersection. Moreover, as the memory is represented as the output, the memory cannot be simultaneously the value m and the value m' present. It is thus harder to predict what would be the behavior of such conceptor. In practice, as for the disjunction, when a trigger occurs the output jumps towards the value v to be maintained and then relaxes to another value. However, in that case the value towards which it relaxes is easy to describe, it is always almost zero. The conjunction of constant-memory conceptors acts therefore as C_0.

4 Discussion

Conceptors are powerful tools for performing explicit operations in the latent space of a reservoir even though the composition of such operations remains a difficult task. Using a reservoir model of gated working memory, we have shown another way to enforce a particular memory through the use of ad-hoc conceptors. These constant-memory conceptors therefore provide a synaptic form to the memory, and we have shown how they can be used to stabilize or discretize the memory. However, the effect of conceptors composition is counter-intuitive and largely differs from what we would naturally expect.

For instance, we have seen that the linear interpolation of constant-memory conceptors does not create another constant-memory conceptor, or at least it does not correspond to one we would have computed. The reason being that the space of constant-memory conceptors is not a straight line. Hence, they cannot be linearly interpolated: a mere linear combination of constant-memory conceptors could not lead

to another constant-memory conceptor. Nevertheless, we have shown empirically that in all scenarios a linear combination of two constant-memory conceptors lead to a value that is maintained. However, this new memory is slightly oscillating around the combination of the constant values (see Figure 6). This oscillation being a direct consequence of the perturbation of the system (i.e. the input).

Moreover, the disjunction of constant-memory conceptors gives an example of functional conceptor. However, it does not implement what we expected. In the case of two conceptors with two constant values v_1 and v_2 such that $0 \leq v_1 \leq v_2$, we would expect that the disjunction encodes the two values simultaneously. More specifically, we expected such disjunction to implement a choice function (i.e. a conditional assignment) between the two values stored in each conceptor. Instead, the disjunction implements another conditional assignment, that does not converge towards v_1 but only towards 0 or v_2 depending on the given input value. To some extent, v_1 and v_2 influence the disjunction with different qualitative roles. Moreover, in the low rank case (i.e. when the recurrent weights are the sum of a low rank matrix and a random perturbation) only the largest fixed points can be stable (Schuessler et al., 2020). Therefore, we can hypothesize that in the general case of a disjunction of $n > 2$ constant-memory conceptors, only the extreme value matter in the composite conceptor.

Even though our results are preliminary and will require more work to fully characterize how operations can be composed intuitively, this work opens the door to another form of working memory: a procedural (or functional) working memory. Instead of temporarily memorizing declarative information, this kind of working memory would be able to memorize procedural information (e.g. how a task should be performed, which processes should be applied, etc.). For instance, imagine you are given some instructions which are to sum up a series of numbers. In order to complete this task, it is necessary to keep track of the current sum (e.g. in a classical short-term declarative working memory) that needs to be updated each time a new number is given. However, it is also necessary to remember the preliminary instruction (i.e summing up) in another form of working memory that needs to span the whole experiment and which is procedural in nature. This procedural nature makes this working memory quite peculiar because instead of memorizing a given information, it needs to memorize a procedure – here, a sequence of operations depending on the context – that needs to be applied each time an input is given. It is not yet clear how such memory could be encoded in the brain (e.g. sustained activity, dynamic activity, transient weights) and we think conceptors might be key in answering such a question, but more experimental and theoretical work will be needed. Similar conceptors than the ones we computed with our gated working memory reservoir model are likely to be computed with other working memory models (Lim and Goldman, 2013; Nachstedt and Tetzlaff, 2017; Bouchacourt and Buschman, 2019): it would be interesting to see whether the functional conceptors obtained are similar (i.e. our results would be generally applicable), or on the contrary, if differences occur.

Compliance with Ethical Standards

Authors declare they have no conflict of interest. Ethical approval: This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bao, Jiao et al. (2016). "Action recognition based on conceptors of skeleton joint trajectories." In: *Rev. Fac. Ing* 31.4, pp. 11–22.

Bartlett, Madeleine et al. (2019). *Recognizing Human Internal States: A Conceptor-Based Approach.* arXiv: 1909.04747 [cs.HC].

Bouchacourt, Flora and Timothy J. Buschman (July 2019). "A Flexible Model of Working Memory." In: *Neuron* 103.1, 147–160.e8. DOI: 10.1016/j.neuron.2019.04.020.

Brock, Andrew et al. (2016). "Neural photo editing with introspective adversarial networks." In: *arXiv preprint arXiv:1609.07093*.

Cho, Kyunghyun et al. (Oct. 2014). "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1724–1734. URL: http://www.aclweb.org/anthology/D14-1179.

DePasquale, Brian et al. (Feb. 2018). "full-FORCE: A target-based method for training recurrent networks." In: *PLOS ONE* 13.2. Ed. by Maurice J. Chacron, e0191527. DOI: 10.1371/journal.pone.0191527. URL: https://doi.org/10.1371/journal.pone.0191527.

Gast, Richard et al. (Apr. 2017). "Encoding and Decoding Dynamic Sensory Signals with Recurrent Neural Networks: An Application of Conceptors to Birdsongs." In: *BioRxiv*. DOI: 10.1101/131052. URL: https://doi.org/10.1101/131052.

He, Xu and Herbert Jaeger (2018). "Overcoming Catastrophic Interference using Conceptor-Aided Backpropagation." In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=B1al7jg0b.

Jaeger, Herbert (Jan. 2001). *The "echo state" approach to analysing and training recurrent neural networks*. Tech. rep. 148. Bonn, Germany: German National Research Center for Information Technology GMD.

– (2014). "Controlling recurrent neural networks by conceptors." In: *arXiv preprint arXiv:1403.3369*.

– (2017). "Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns." In: *Journal of Machine Learning Research* 18.13, pp. 1–43.

Lim, Sukbin and Mark S Goldman (Aug. 2013). "Balanced cortical microcircuitry for maintaining information in working memory." In: *Nature Neuroscience* 16.9, pp. 1306–1314. DOI: 10.1038/nn.3492.

Liu, Tianlin, Lyle Ungar, and João Sedoc (2019). "Continual Learning for Sentence Representations Using Conceptors." In: *CoRR* abs/1904.09187. arXiv: 1904.09187. URL: http://arxiv.org/abs/1904.09187.

Manohar, Sanjay G. et al. (June 2019). "Neural mechanisms of attending to items in working memory." In: *Neuroscience & Biobehavioral Reviews* 101, pp. 1–12. DOI: 10.1016/j.neubiorev.2019.03.017.

Masse, Nicolas Y. et al. (June 2019). "Circuit mechanisms for the maintenance and manipulation of information in working memory." In: *Nature Neuroscience* 22.7, pp. 1159–1167. DOI: 10.1038/s41593-019-0414-3.

Mikolov, T. et al. (2013). "Distributed representations of words and phrases and their compositionality." In: *Proc. of NIPS*, pp. 3111–3119.

Mongillo, G., O. Barak, and M. Tsodyks (Mar. 2008). "Synaptic Theory of Working Memory." In: *Science* 319.5869, pp. 1543–1546. DOI: 10.1126/science.1150769.

Mossakowski, T., R. Diaconescu, and M. Glauer (2019). "Towards logics for neural conceptors." In: *J of Applied Logics* 6.4, pp. 725–744.

Nachstedt, Timo and Christian Tetzlaff (May 2017). "Working Memory Requires a Combination of Transient and Attractor-Dominated Dynamics to Process Unreliably Timed Inputs." In: *Scientific Reports* 7.1. DOI: 10.1038/s41598-017-02471-z.

Schuessler, Friedrich et al. (2020). "Dynamics of random recurrent networks with correlated low-rank structure." In: *Physical Review Research* 2.1, p. 013111.

Stokes, Mark G. (July 2015). "'Activity-silent' working memory in prefrontal cortex: a dynamic coding framework." In: *Trends in Cognitive Sciences* 19.7, pp. 394–405. DOI: 10.1016/j.tics.2015.05.004.

Strock, Anthony, Xavier Hinaut, and Nicolas P. Rougier (Jan. 2020). "A Robust Model of Gated Working Memory." In: *Neural Computation* 32.1, pp. 153–181. DOI: 10.1162/neco_a_01249.